GWD-I (Informational)

R. Aydt, University of Illinois at Urbana-Champaign
D. Gunter, Lawrence Berkeley National Laboratory
W. Smith, NASA Ames Research Center
B. Tierney, Lawrence Berkeley National Laboratory
V. Taylor, Northwestern University

GGF Performance Working Group

February 2001
Revised 27-August-2002

A Simple Case Study of a Grid Performance System

Status of this Memo

This memo provides information to the Grid community regarding a simple performance monitoring scenario and an abstract implementation of a Grid performance system based on the Grid Monitoring Architecture (GMA) being developed by the Global Grid Forum Performance Working Group. Distribution is unlimited.

1. Abstract

The Global Grid Forum Performance Working Group has been developing a Grid Monitoring Architecture (GMA) that outlines one possible approach for monitoring resources and applications in a Grid environment. This document presents a simple case study of a Grid monitoring system based on the GMA. It describes how the various system components interact for a very basic monitoring scenario, and it introduces the terminology and concepts presented in greater detail in other Working Group documents. This document is intended to provide a basis for further discussion and prototype implementations of Grid monitoring systems.

aydt@uiuc.edu

R. Aydt, University of Illinois at Urbana-Champaign
D. Gunter, Lawrence Berkeley National Laboratory
W. Smith, NASA Ames Research Center
B. Tierney, Lawrence Berkeley National Laboratory
V. Taylor, Northwestern University

GGF Performance Working Group

February 2001
Revised 27-August-2002

Table of Contents

aydt@uiuc.edu

2. Introduction

This document presents a simple case study of a Grid performance system based on the Grid Monitoring Architecture (GMA) [1] being developed by the Global Grid Forum Performance Working Group. It describes how the various system components would interact for a very basic monitoring scenario, and it introduces the terminology and concepts presented in greater detail in other working group documents.

Prototype implementations of this basic scenario can be built to explore the feasibility of the proposed architecture and to expose possible shortcomings. Once the simple case is understood and agreed upon, complexities can be added incrementally as warranted by cases not addressed in the most basic implementation described here.

Some open issues and complex requirements that came up during the discussions of the simple scenario are briefly noted, but no attempt is made to address them in this document. We also do not address scalability, performance, or fault tolerance in this document. In the simple case presented here, we assume that all components have the necessary authorization to interact in the manner described. Clearly, authentication and authorization are two important considerations in a real implementation of any Grid performance system.

3. Scenario

Ten workstations (*ws1–ws10)* are used as desktop systems by local users and are also available as compute engines for Grid applications. A monitor is running on each of these workstations to measure the CPU load every 30 seconds. The CPU load measurements are all forwarded to a central server machine (*srvr)* on the same local area network as the workstations. A process on the server makes the load information available to systems not located on the local network.

A system administrator for the *ws* workstations monitors the loads from her machine, *adminsys,* to ensure that no problems exist. Her machine, *adminsys,* is on a different network than the *ws* workstations.

Further, all of the load measurements are being archived by an archiving service on the machine *archivsys.* The archival data is used by another program not discussed in the simple case study to analyze daily system load patterns and to identify time periods when the workstations are heavily used, so that backups will not be scheduled during those times.

Figures depicting the components involved in the scenario and the interactions between them (described in upcoming sections) can be found in the Appendix (page 11).

4. Terminology

In this section we define some of the basic terms used by the Global Grid Forum Performance Working Group and relate them to the simple case study presented in this document.

4.1 Event, Event Type, and Event Data

An *event*[*] is a structure containing one or more items of data that relate to one or more resources. Every event has an associated e*vent type* that uniquely identifies the structure for that

[*] *Events,* as defined and used in this document, are implicitly *performance events.* We make no attempt to define or discuss other types of events.

particular event. The term *event data* refers to one or more of the items of data making up an event.

In the scenario described above, the machines *adminsys* and *archivsys* will receive events of type CPU_LOAD describing the load for systems *ws1* through *ws10.* Depending on the implementation, a single event may or may not contain information for all of the systems. In the implementation outlined below, an event contains the CPU load information for a single system.

4.2 Event Schema

An *event schema* describes the structure for a particular event.

In the basic scenario described in this document, a schema will be defined for the CPU_LOAD event type.

Note that the data model for events has not yet been fully defined. In this document we restrict ourselves to a minimal data model of a named structure with one or more elements consisting of a tuple of (name, data type, value).

4.3 Producer and Producer Interface

A *producer* makes event data available to other components that are part of the Grid Monitoring Architecture. A given process or component may have multiple *producer interfaces,* each acting independently and providing event data. The term *producer* is used interchangeably, and inexactly, to refer both to a single producer interface and to a process or component that contains at least one producer interface.

A producer (interface) speaks a standard protocol and generates event data in a standard format. Several standard protocols and formats will probably be defined within the GMA, and a single producer may support multiple options. A producer may not be the originating source of the data–that source may or may not speak the same protocol and use the same event data format.

In our scenario, a process on *srvr* is a producer and makes event data available to other components in the Grid performance system being described. We assume that monitoring processes on *ws1* through *ws10* are the originating sources of the measurement data, but they are not publishing their results via GMA, and therefore are not producers.

4.4 Consumer and Consumer Interface

A *consumer* in the Grid Monitoring Architecture receives event data from a producer. A given process or component may have multiple *consumer interfaces,* each acting independently and receiving event data. The term *consumer* is used interchangeably, and inexactly, to refer both to a single consumer interface and to a process or component that contains at least one consumer interface. A consumer (interface) speaks a standard protocol and expects the event data to be in a standard format.

In the basic scenario described, processes on *adminsys* and *archivsys* are consumers of the event data produced by *srvr.* The *adminsys* consumer process will monitor the per-host CPU_LOAD measurements. The *archivsys* consumer process will write the event data to disk for later examination.

Note that although the *archivesys* process described in this scenario has a consumer interface, the same process may also have a producer interface that is active when data is extracted from the archive.

4.5 Directory Service

A *directory service* is a searchable component in the Grid Monitoring Architecture used to store and forward information that is of general interest to other components in the system. The directory service can be queried through a variety of search mechanisms and returns information matching the specified selection criteria. The directory service may in practice be implemented as a set of distributed, interconnected individual directory services under the control of different organizations.

In the GMA, several distinct types of information will be stored in the directory service, and we refer to the directories for each information type by a unique name. The actual implementation may place all entries in a single directory service, but conceptually we believe it is easiest to think of them as independent directories. Here we define only those directories that are necessary to implement the basic scenario.

4.5.1 Event Type Directory

The *Event Type Directory* contains event schema for the various events in the system. The Event Type Directory does *not* contain actual events. For each event type there will be one schema in the Event Type Directory–within the system all events of the same type must have the same structure.

The Event Type Directory can be searched by event type. It can also be searched by event element name, for example, “return all the event types that contain an element named *measurement*.”

To support the basic scenario described, the CPU_LOAD schema must be included in the Event Type Directory.

4.5.2 Event Producer Directory

The *Event Producer Directory* contains information about producers and the event types they provide. This information is registered by the producer, either dynamically or when the producer is started.

All producer information in the Event Producer Dictionary is structured according to an *Event Producer Schema.* In contrast to the Event Type Directory, which contains the event schema but not the actual events, the Event Producer Directory *does* contain the actual producer information records and not just the schema for those records.

Consumers use the Event Producer Directory to locate producers of events they are interested in receiving. A consumer might want to search for producers in the Event Producer Directory in various ways, including by event type, by producer, or by host where the measurement originated. The set of search keys that should be supported is an open question.

For the basic scenario outlined in this document, the Event Producer Directory will contain one or more entries indicating that CPU_LOAD event data for *ws1* through *ws10* is available from a producer on *srvr.*

4.5.3 Event Consumer Directory

The *Event Consumer Directory* contains information about consumers, the event types they accept, and the services they provide.

All consumer information in the Event Consumer Dictionary is structured according to an *Event Consumer Schema.* As with the Event Producer Directory, the Event Consumer Directory *does* contain the actual consumer information records and not just the schema for those records.

Producers use the Event Consumer Directory to locate consumers that provide services of interest, or to find information on supported control and data protocols for known consumers. As with the Event Producer Directory, the set of search keys that should be supported for the Event Consumer Directory remains an open question.

To support the basic scenario described in this document, the archival process on *archivsys* will register with the Event Consumer Directory as a consumer that accepts all event types and provides an archival service. This registration will allow the producer process on *srvr* to locate the archival service. In another possible implementation, the archive consumer would not register its service, but instead locate and initiate a connection to the *srvr* producer.

5. Implementation

In this section we describe, at a fairly high level, the steps necessary to implement the basic scenario with the Grid Monitoring Architecture. Our objective is to give the reader a clear idea of how the GMA components cooperate and to provide a framework from which prototype implementations can be developed to test various protocols and formats.

5.1 Event Schema and Event Type Directory

To implement the basic Grid performance system described, we must first define the event schema for the CPU_LOAD event. This schema will be stored in the Event Type Directory where it can be located and used to interpret data values in CPU_LOAD events. We use a representation-independent format to define the schema (see Fig. 1).

Event Type		**Event Description**
CPU_LOAD		CPU load measurement for a single host
Element Name	**Element Data Type**	**Element Description**
measurement	double	measured CPU load
hostname	string	host where measurement was taken
timestamp	IETF timestamp[2]	time measurement was taken

Figure 1: CPU_LOAD Schema

As defined, a *CPU_LOAD* event has three data elements that contain the CPU load measurement, the host the measurement relates to, and the time the measurement was made.

5.2 Event Producer Directory

The next step in the implementation process is for the producer, *srvr,* to add entries to the Event Producer Directory, advertising that it will provide CPU_LOAD event data for *ws1, ws2, … ,ws10.*

We have not yet reached a consensus on the contents of the Event Producer Directory entries, that is, the Event Producer Schema has not yet been set. Further discussion and experimentation are required to correctly identify an appropriate Event Producer Schema. The version presented here should not be interpreted as a standard.

For the purpose of this simple case study we list in Figure 2 the type of information that might be included in the Event Producer Directory entries. One Event Producer Directory entry is shown, that for the CPU load data from *ws1.* Similar entries will exist for *ws2* through *ws10*.

Field Name	Value
Producer_URL	srvr:portXX
Event_Type	CPU_LOAD
Host	ws1
Service	basic
Parameters	NONE
Filters	NONE
Access_Control	OPEN
Control_Protocol	SOAP_HTTP, SOAP_TCP, JAVA_RMI
Data_Protocol	SOAP_HTTP, SOAP_UDP

Figure 2: Event Producer Directory Entries

In the simple case study presented in this document, the consumer on *adminsys* is interested in CPU_LOAD data for any of the *ws* machines. To support this scenario, the Event Producer Directory will be searched for entries with an *Event_Type* of "CPU_LOAD" and a *Host* of "ws1" through "ws10." The *Producer_URL* field specifies where to contact the producer to receive events of interest.

The remaining Event Producer Directory fields are not explicitly used in this simple case study but are included to show possible extensions. *Service* could be used to indicate the types of queries that are supported by the producer interface. *Parameters* could be used to indicate that the producer would allow the consumer to control some producer variables, such as frequency of event record transmission. The *Filters* field could be used to indicate that the producer has some built-in filtering capabilities, such as sliding window average computations. The *Access_Control* field is intended to provide different levels of access to the event data that is being produced – for example, make data available only to consumers within the same organization or make data available to anyone. Additional security-related fields such as digital signature would also be stored in the event producer directory.

The *Control_Protocol* and *Data_Protocol* fields could be used to specify which of several standard wire protocols the producer understands. These fields allow for different control and data transport protocols. Sample protocols include SOAP_HTTP[3], SOAP_TCP, SOAP_UDP, and JAVA_RMI[4]. A consumer may be fluent in a limited set of the possible protocols and consequently would consider connecting only to producers that "speak" those protocols.

5.3 Event Consumer Directory

Another step in the implementation process is for the archiving consumer on *archsys* to advertise its existence, allowing the *srvr* producer to locate it. As mentioned earlier, a different implementation may have an archive process that does not register but instead initiates the connection to the producer process.

As with the Event Producer Directory entries, the Event Consumer Schema describing the contents of the Event Consumer Directory entries has not yet been set. For the purpose of this simple case study we show the type of information that might be included in the Event Consumer Directory entries. An entry for the archiving consumer on *archsys* is shown in Figure 3.

Field Name	Value
Consumer_URL	archsys:portYYY
Event_Type	*
Service	archive
Parameters	NONE
Access_Control	Producer=*.mydomain.edu
Control_Protocol	SOAP_TCP
Data_Protocol	SOAP_UDP

Figure 3: Event Consumer Directory Entry

The *Consumer_URL* field specifies where to contact the consumer process, the *Event_Type* field indicates the types of events the consumer is willing to accept, and the *Service* field shows the service or services the consumer provides. The other fields correspond to like-named fields in the Event Producer Schema. An asterisk indicates a wildcard.

5.4 Consumer/Producer Communication Established

Now that the directory service contains the event type schema, event producer information, and event consumer information, the Grid performance system is ready to share measurement information from resources in one part of the Grid with processes running on other systems in the Grid.

In particular, for our simple case study the monitoring tool running on *adminsys* posts a query to the Event Producer Directory requesting any CPU_LOAD events for machines *ws1* through *ws10.* The query returns ten matches, all with the same *Producer_URL* contact values. Using one of the *Control_Protocol*s retrieved from the Event Producer Directory, the monitoring tool on *adminsys* connects to the producer process at *srvr:portXX* and subscribes to the CPU_LOAD events for *ws1, ws2, … ,ws10.*

After starting up, the producer process on *srvr* queries the Event Consumer Directory to find a consumer that offers archival services for CPU_LOAD events whose protocols are compatible with those of the producer. Assuming *srvr* is in "mydomain.edu," the entry for the archival service on *archivsys* is returned. At this point, the producer process on *srvr* contacts the archival consumer process on *archivsys* and initiates a subscription to the producer's CPU_LOAD events for *ws1* through *ws10*.

5.5 Event Data Sent by Producer to Consumers

Once the subscriptions are in place, the producer sends CPU_LOAD event data to the consumers until the subscriptions are canceled by the consumer or timed out.

The event data is sent by using one of the protocols advertised in the Event Producer Directory. If the producer or consumer advertised that it understands multiple data formats, then the particular format is specified or negotiated in the connection process.

Sample event data encoded in XML[5] is shown here, with white space added for readability:

```
<CPU_LOAD>
   <measurement> 30.09 </measurement>
   <hostname> ws1 </hostname>
   <timestamp> 2001-01-30T20:33:05.003Z</timestamp>
   <producer> http://srvr.mydomain.edu/producerXX </producer>
</CPU_LOAD>
```

```
<CPU_LOAD>
   <measurement> 22.98 </measurement>
   <hostname> ws9 </hostname>
   <timestamp> 2001-01-30T20:34:15.07Z</timestamp>
   <producer> http://srvr.mydomain.edu/producerXX </producer>
</CPU_LOAD>
```

The monitoring tool receives the event data and updates the display for each host with the appropriate measurements. The archiving service receives the event data and writes it to the archive for later analysis by the backup-scheduling program.

6. Summary

We have described a very basic performance monitoring scenario in a Grid environment, defined terms used within the Global Grid Forum Performance Working Group, related those to the scenario, and outlined at a fairly high level how the scenario could be implemented with the components defined in the Grid Monitoring Architecture. This basic scenario ignores many important and complex issues that are critical to a fully functional Grid performance system; our objective is to present basic concepts and to provide a starting point for discussion and prototype implementation experiments.

7. Open Issues

Many aspects of the GMA are not yet fully defined, including the event data model, directory service entries and search procedures, security mechanisms, and protocols.

Readers are encouraged to visit the GGF Performance Area Web site, which is accessible from the main GGF Web site located at http://www.gridforum.org, to view the latest GMA document and related proposals and prototypes. Interested parties are also welcome to participate in ongoing discussions regarding the GMA by attending GGF meeting and contributing to the Performance Area mailing list.

8. Security Considerations

The document acknowledges that authorization and authentication are critical elements of a Grid monitoring system but makes no attempt to address how the described system components would implement these security features.

Author Information

Ruth A. Aydt University of Illinois at Urbana-Champaign aydt@uiuc.edu ph +1-217-333-8924 fax +1-217-244-7396	Dan Gunter Lawrence Berkeley National Laboratory dkgunter@lbl.gov	Warren Smith NASA Ames Research Center wwsmith@arc.nasa.gov
Brian L. Tierney Lawrence Berkeley National Laboratory bltierney@lbl.gov	Valerie Taylor Northwestern University taylor@ece.nwu.edu	

Glossary

GMA	Grid Monitoring Architecture, as defined by the Global Grid Forum Performance Working Group.

Acknowledgements

Thanks to Darcy Quesnel, who participated in the initial SC2000 hallway discussions of the simple case study that led to this document, and to Rich Wolski, who offered valuable comments during the final revision phase. Thanks also to all members of the Performance Working Group who contributed to the initial and ongoing GMA discussions.

Ruth Aydt is supported in part by the Department of Energy under contract DOE W-7405-ENG-36 and by the National Science Foundation under Grant No. 9975020. Dan Gunter and Brian Tierney are supported by the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Any opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

Intellectual Property Statement

Full Copyright Notice

Appendix: Component Interaction Figures

Figure 4 shows the interactions between the components of the scenario presented in this document that relate to registration, discovery, and subscription. Figure 5 shows the flow of performance events between the producer (*srvr)* and consumers *(archivsys* and *adminsys).*

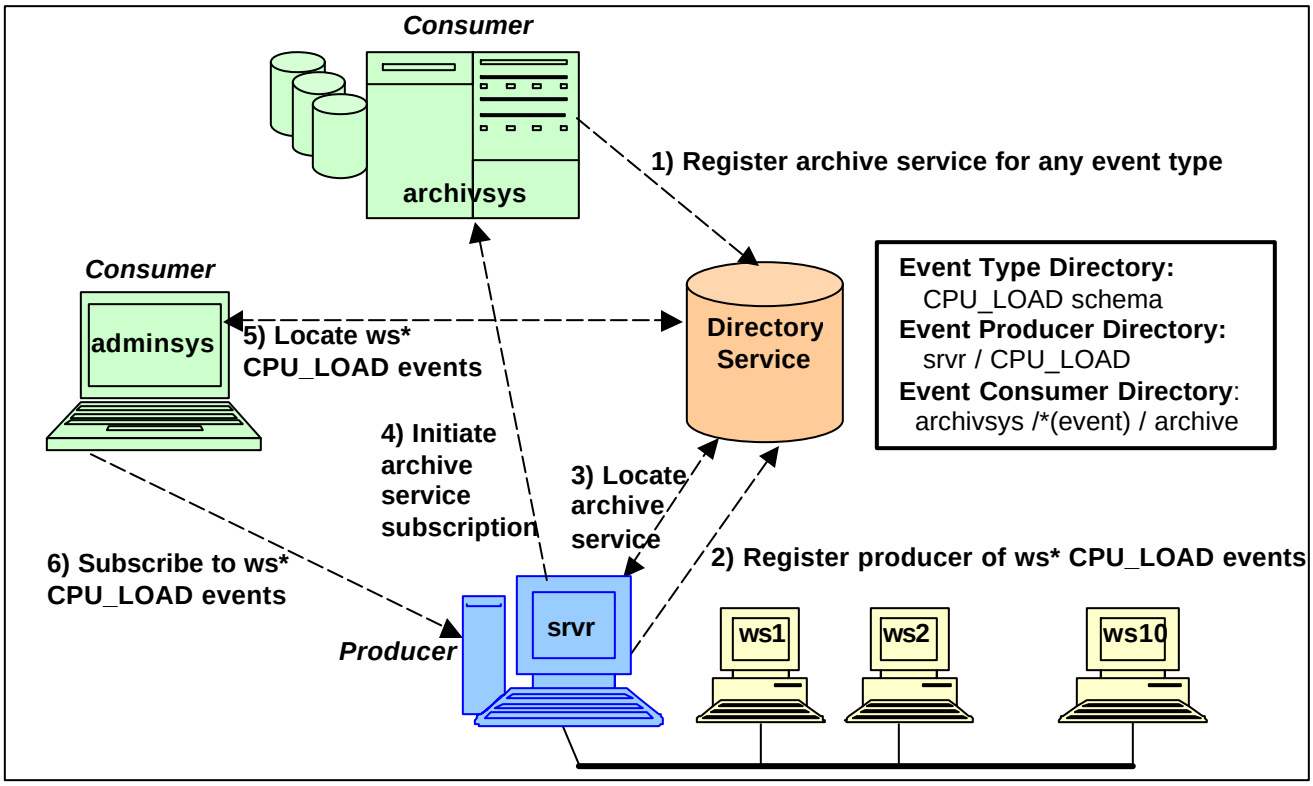

Figure 4: Registration, Discovery, and Subscription Interactions

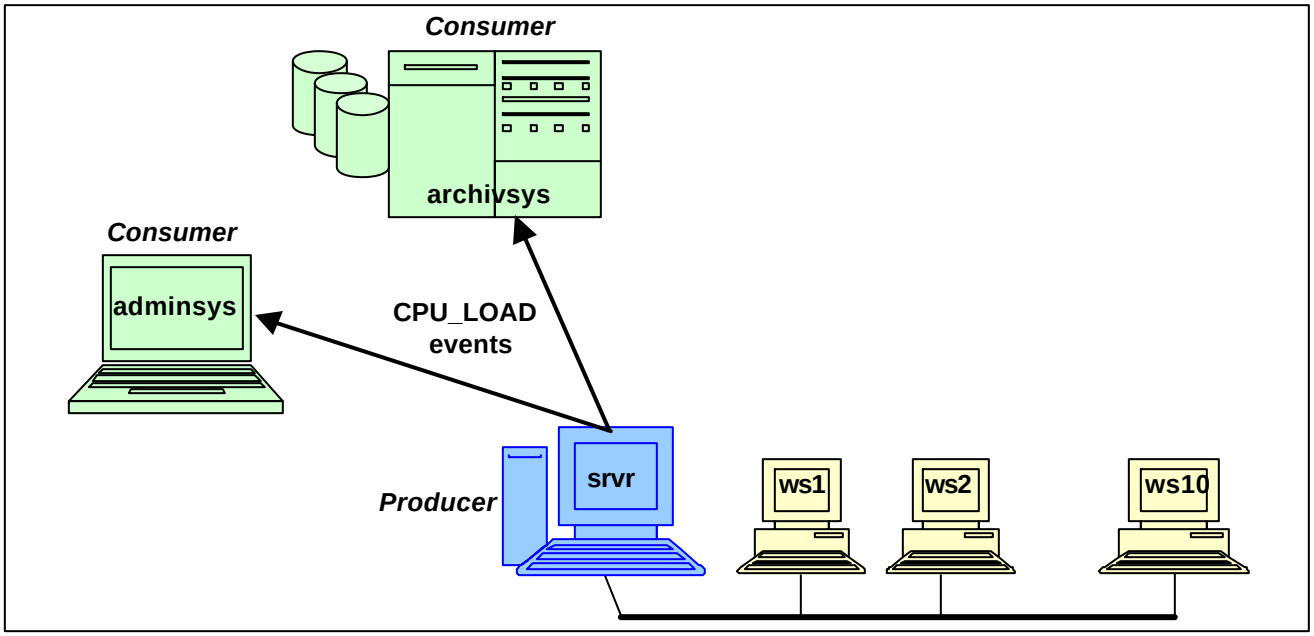

Figure 5: Flow of Performance Events

References

[1] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, R. Wolski. *A Grid Monitoring Architecture.* GGF Document series available from http://www.gridforum.org.

[2] G. Klyne, C. Newman. *Date and Time on the Internet: Timestamps.* IETF Internet-Draft working document. Currently available at http://ietf.org/internet-drafts/draft-ietf-impp-datetime-05.txt

[3] *Simple Object Access Protocol (SOAP) 1.1.* http://www.w3.org/TR/SOAP.

[4] *Java™ Remote Method Invocation (RMI).* http://java.sun.com/products/jdk/rmi.

[5] *XML – Extensible Markup Language.* http://www.w3.org/XML.